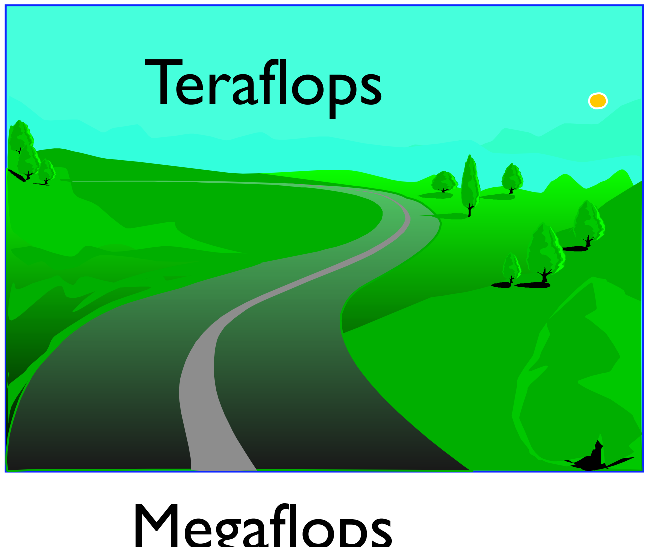
IBM's Advanced Computing Technology Center

Moving Beyond Teraflops Workshop



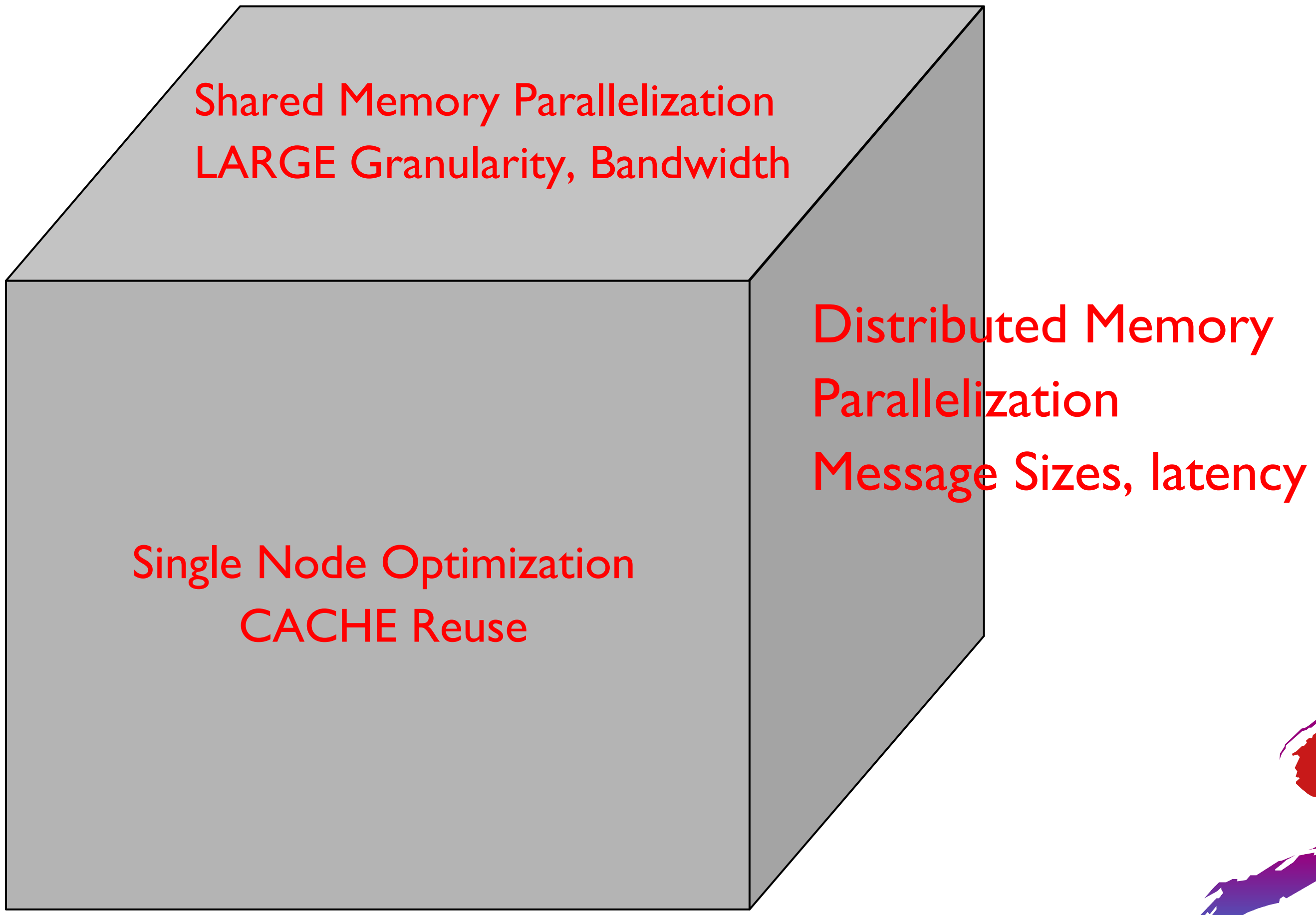
What is ACTC

- The Advanced Computing Technology Center has been established at IBM Research to focus expertise in High Performance Computing and supply the user community with solutions to porting and optimizing their applications

What is ACTC trying to do:

- Assist in taking IBM to leadership in HPC and keep it there
- Hardware roadmap is excellent
- Takes more than that
  - HPC Sofware at IBM isn't of leadership quality
  - HPC Customer Support isn't of leadership quality
- Emulate what Cray Research did in the 80-90s
  - Work with customers to solve their most difficult problems
  - Identify holes in IBM HPC software offerings and plug the holes
  - Form alliance with users
    - SCIENTIFIC USER SUPPORT REQUIRES LONG-TERM RELATIONSHIP

3-Dimensional Optimization



Agenda

- Tuesday Morning
- 9:00-10:30 David Klepacki
  - ACTC Introduction
  - Power3 and Power4 SP Hardware Architecture
    - Winterhawk /Nighthawk
    - Gigaprocessor (GP) Overview
  - Power3 Uni-processor Optimization
    - Cache Utilization
    - HW Prefetch and Algorithmic Prefetching
    - Utilization of the Functional Units

Agenda

- Tuesday Morning
- 10:45 - 12:30 Bob Walkup
  - Useful PSSP and PE commands
  - AIX-based Tools (vmstat...etc., AIX trace facility)
  - Debugging and Performance Tools
    - Useful Tools (Performance Toolbox, Xprofiler)
    - Simple Debugging techniques (pdbx)
    - Compiler Switches and Environment Variables
    - Using MASS and ESSL

Agenda

- All afternoons will be porting and optimizing applications in a workshop environment

Agenda

- Wednesday Morning
- 9:00 - 10:30 David Klepacki
  - Shared Memory Parallel Programming
    - The Pthreads Model
    - A Pthreads template for C
    - Pthreads performance issues

Agenda

- Wednesday Morning
- 10:45 - 12:00 Charles Grassl
  - OpenMP Optimization on SMP Nodes
    - OpenMP
    - Compiler Issues (Switches)
    - Minimizing SMP Overhead
    - Environment Variables

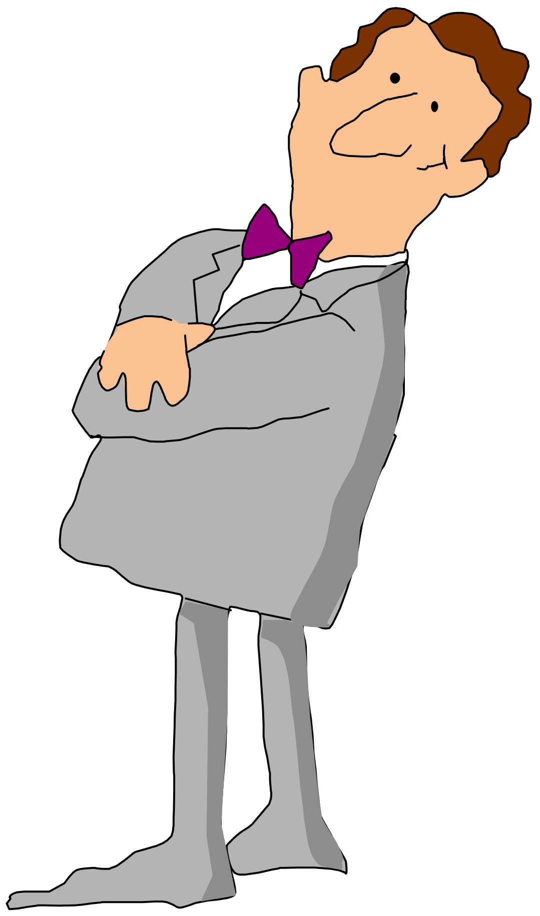
Agenda

- Thursday Morning
- 09:00 - 10:15 Bob Walkup
  - MPI Optimization on the SP
    - Minimizing Message passing overhead
    - MPI Overview and Internals
    - Environment Variables

Agenda

- Thursday Morning
- 10:30 - 12:00   David Klepacki / Charles Grassl
  - MPI + OpenMP on the SP/SMP
    - Combining MPI and OpenMP
    - Performance Considerations
  - The SPMD model

Disclaimer



- Member of IBM Research
- Many designs invented in IBM Research
- Few became product

IBM's Chip technology

- Smaller, less power, cooler, faster
- Limiting factor in future will be power to the chip
  - Logic uses lots of Power
  - Memory uses less power than Logic
  - More memory and less logic on the chip

Superscalar Architectures

- 500 - 2000 MHZ
  - Power 3 - Power 4
- Memory will not keep up with Chip
- More memory on chip
  - Larger Caches on Chip
- More levels of Cache
  - Hide latency to memory
- Main memory further away
  - Don't want to go there

IBM Hardware - Power 3

- Winterhawk I - 200 MHZ (Fall 1998)
- 2-way SMP
- 2 MULT/ADD - 800 MFLOPS
- 64 KB Level I - 5 nsec/3.2 GB/sec
- 4 MB Level 2 - 45 nsec/6.4 GB/sec
- 1.6 GB/S Memory Bandwidth
- 1.6 GFLOPS/Node

- Nighthawk I - 222 MHZ (Summer 1999)
- 8-way SMP
- 2 MULT/ADD - 888 MFLOPS
- 64 KB Level I - 5 nsec/3.2 GB/sec
- 4 MB Level 2 - 45 nsec/6.4 GB/sec
- 14 GB/S Memory Bandwidth
- 7.1 GFLOPS/Node

Power3 Architecture

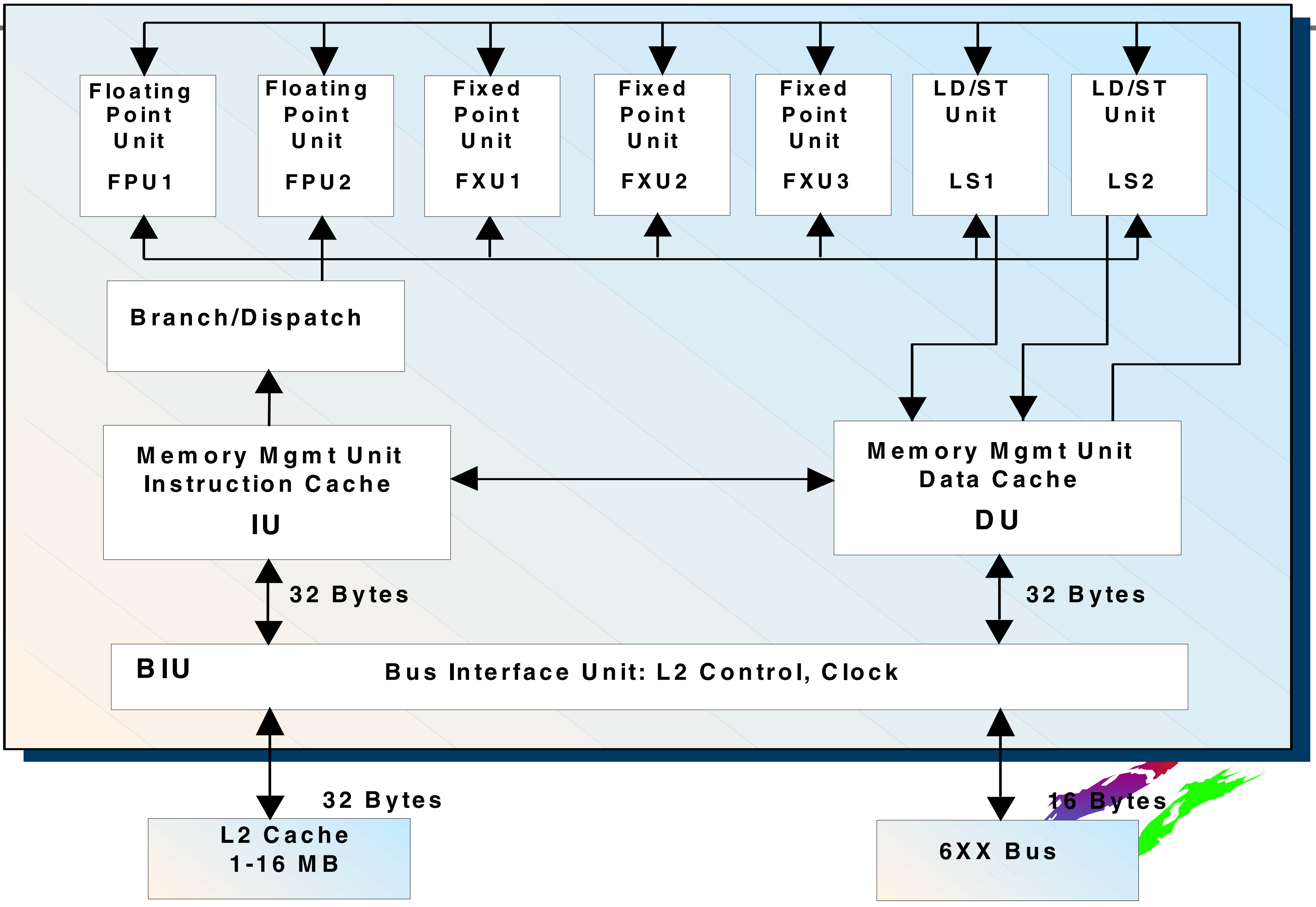
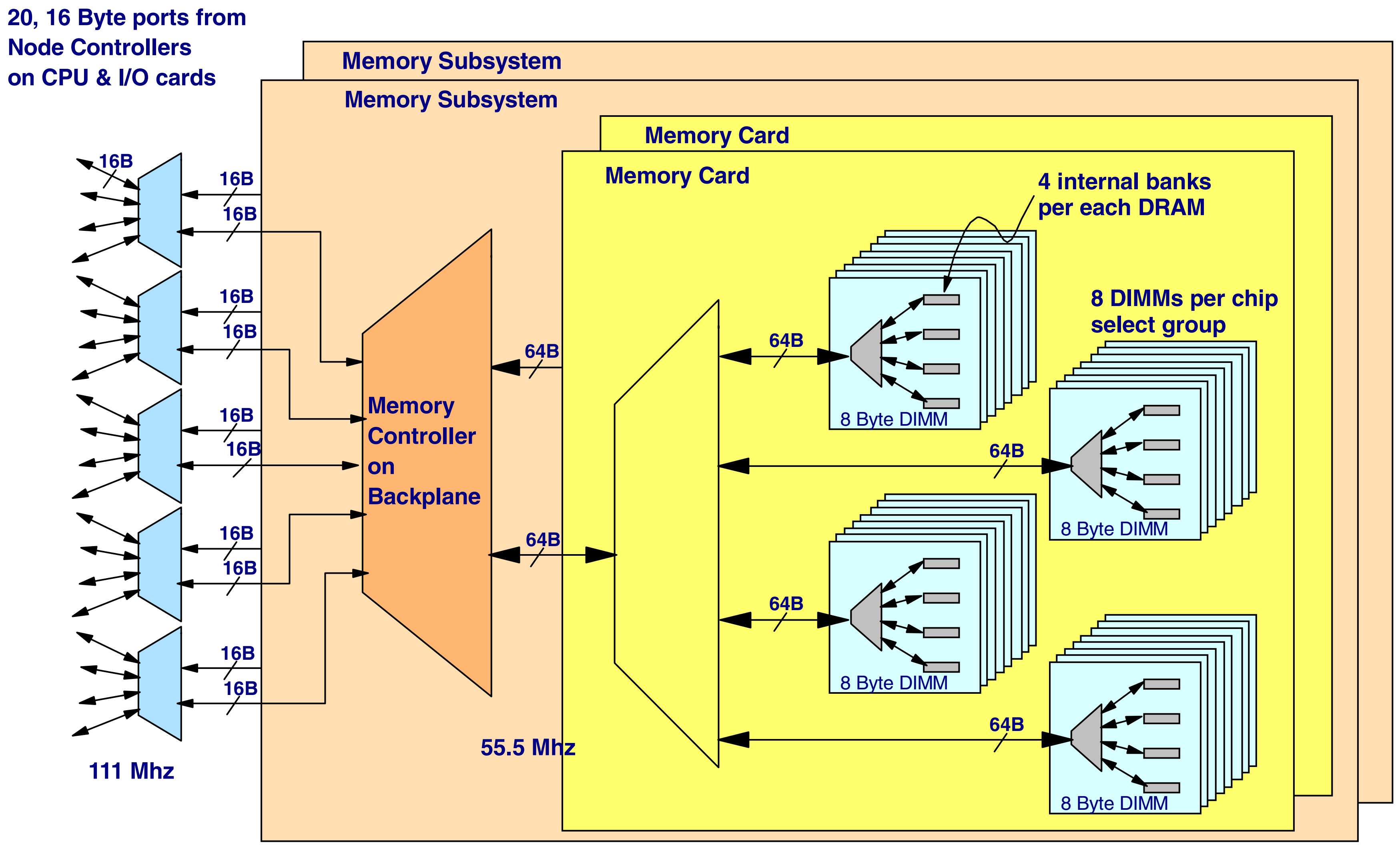


Figure 1. POWER3 Block Diagram

AS/RS

Nighthawk Memory Physical Hierarchy

14.2 GBytes/sec



20, 16 Byte ports from Node Controllers on CPU & I/O cards
Memory Subsystem
Memory Subsystem
Memory Card
Memory Card
4 internal banks per each DRAM
8 DIMMs per chip select group
16B
64B
Memory Controller on Backplane
8 Byte DIMM
111 Mhz
55.5 Mhz

Power 3 Level 1 Cache

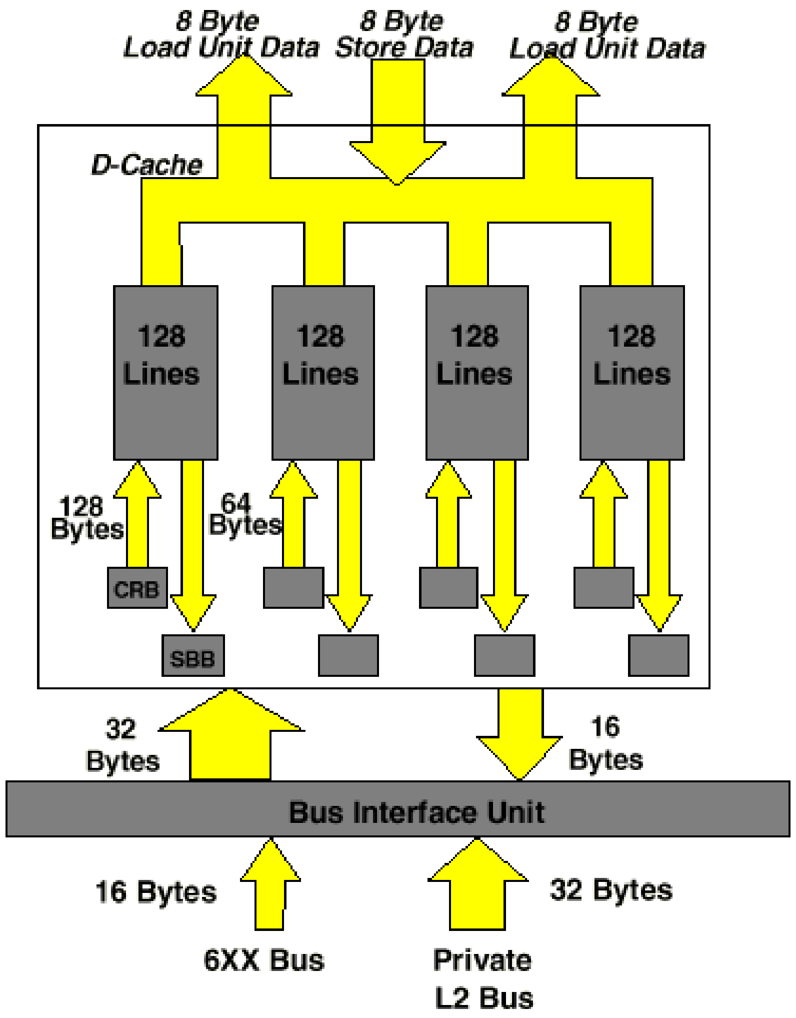


Figure 4. High Bandwidth Interface

One cycle cache access
- One cycle load-to-use

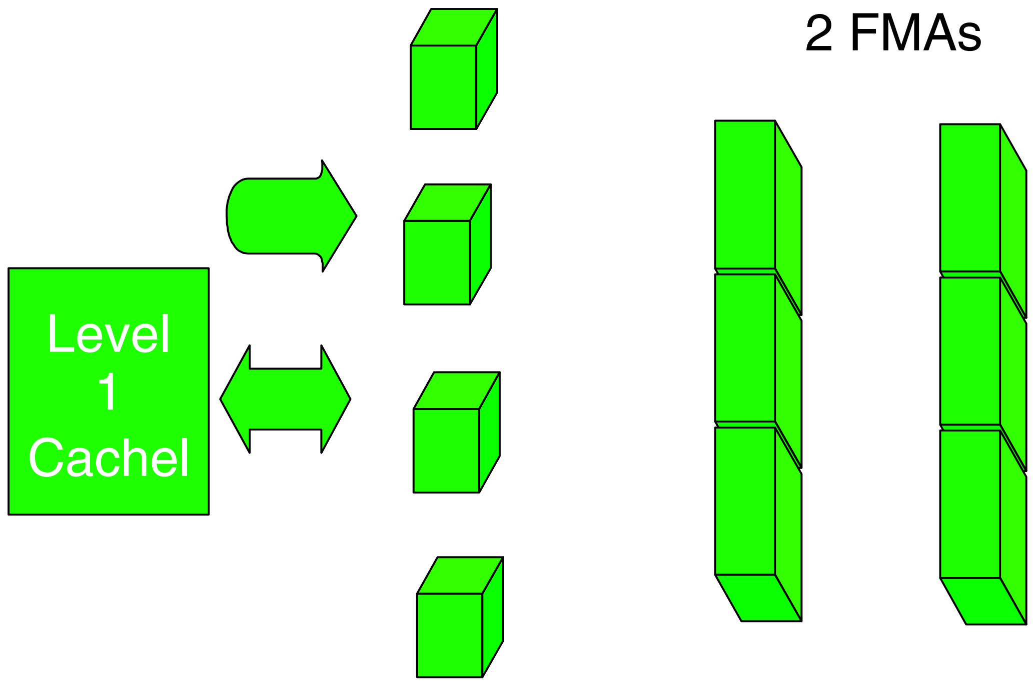
Two reads,one write and one reload per cycle

64KB data cache
- 128-way set associative
- 8-way interleaved
  - 4-way by line
  - 2-way by double word

Four 128-byte cache reload buffers

Four 128-byte cache store back buffers

Power 3 FMA



2 Loads or
1 Load/Store
from
L1 to Registers

Registers
40 Real (24 + 16)
64 Virtual (32 + 32)

Each FMA can
produce a Mult&Add
each cycle

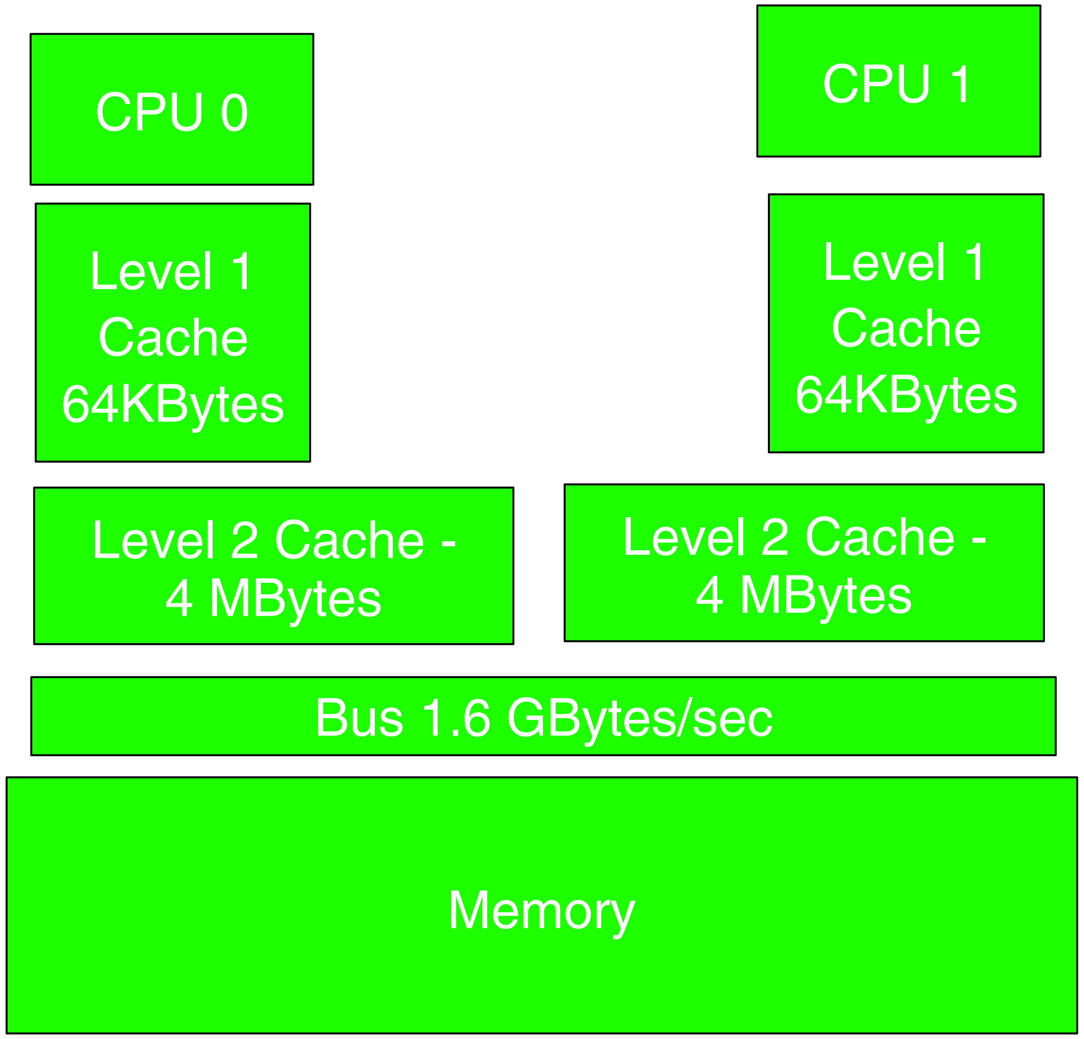
Prefetch Pipes

- 10 Prefetch Registers
- 4 Prefetch Pipes
- Consider:

```
DO i = k,l,j
a(i) = a(i) + 1.0
END DO
```

- If address of a(k) is in the first part of cache line, a forward prefetch is predicted, if the address is in the second half of the cache line a backward prefetch is predicted
- If the address a a(k+j) agrees with prediction, the next or previous cache line will be prefetched

Memory Architecture of the Winterhawk I

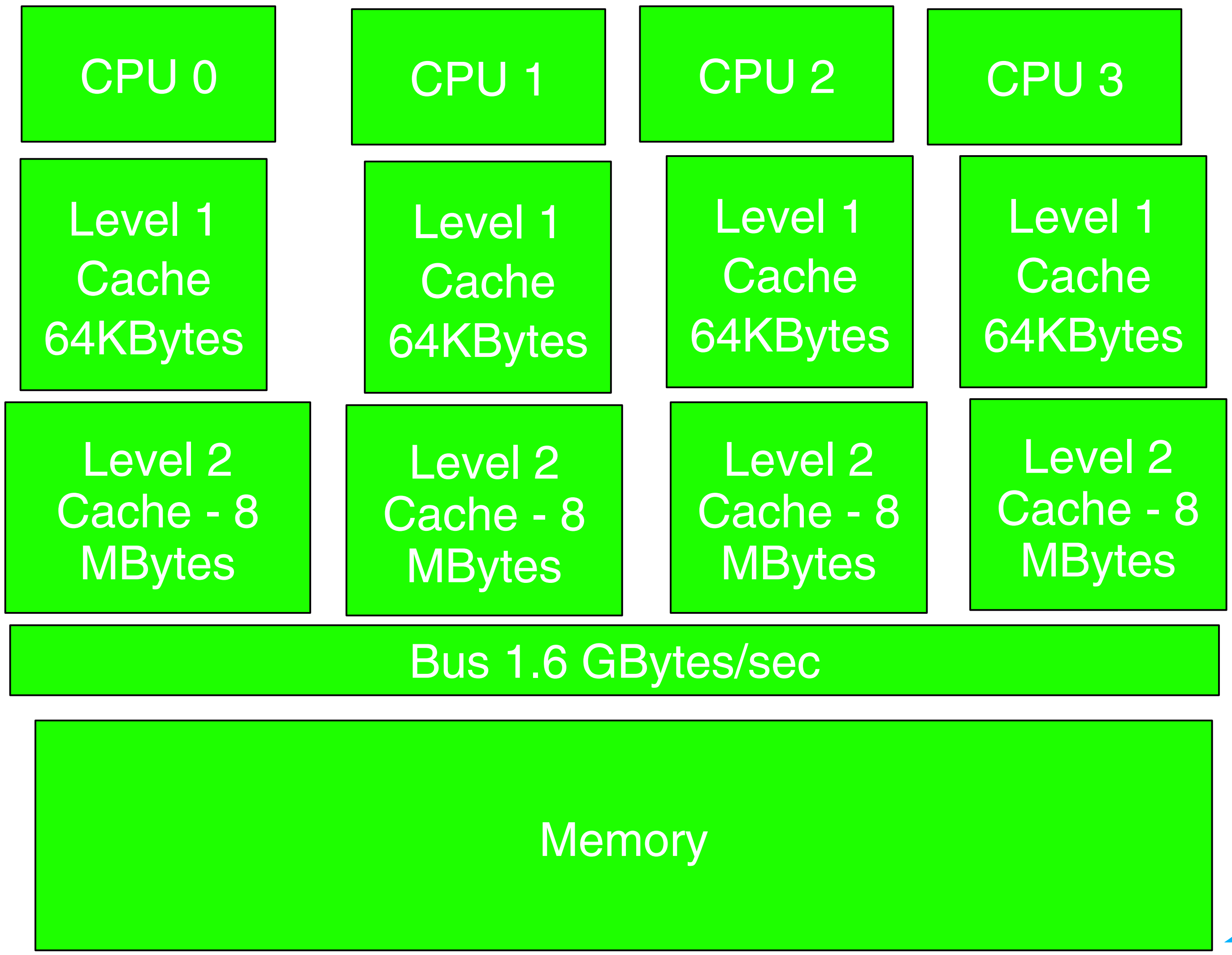


Memory Access times on the Winterhawk 1 Power 3

Access	Latency cycles	Inter Width Bits	Clock MHz	Band-width Bytes/cycle	Band-width GB/sec
Load Register from L1	1	128	200	2*8	3.2
Store Register from L1	1	64	200	8	1.6
Load/Store L1 from/to L2	9	256	200	4*8	6.4
Load/Store L1 from/to Memory	35	128	100	2*8	1.6

Accessing an operand from L1 is at least 35 times faster than from memory

Memory Architecture of the Winterhawk 2



Memory Access times on the Nighthawk I Power 3

Access	Latency clocks	Inter Width Bits	Clock MHz	Band-width Bytes/cycle	Band-width GB/sec
Load Register from L1	1	128	200	2*8	3.2
Store Register from L1	1	64	200	8	1.6
Load/Store L1 from/to L2	9	256	200	4*8	6.4
Load/Store L1 from/to Memory	60	128	100	2*8	1.6

Accessing an operand from L1 is at least 60 times faster than from memory

IBM Hardware - Power 3 + COPPER

- Winterhawk II - ~375 MHZ (2000)
- 4-way SMP
- 2 MULT/ADD - 1400 MFLOPS
- 64 KB Level 1 - 5 nsec/3.2 GB/sec
- 8 MB Level 2 - 45 nsec/6.4 GB/sec
- 1.6 GB/S Memory Bandwidth
- 5.6 GFLOPS/Node

- Nighthawk II - ~375 MHZ (2000 - ASCI)
- 16-way SMP
- 2 MULT/ADD - 1400 MFLOPS
- 64 KB Level 1 - 5 nsec/3.2 GB/sec
- 8 MB Level 2 - 45 nsec/6.4 GB/sec
- 14 GB/S Memory Bandwidth
- 22.4 GFLOPS/Node

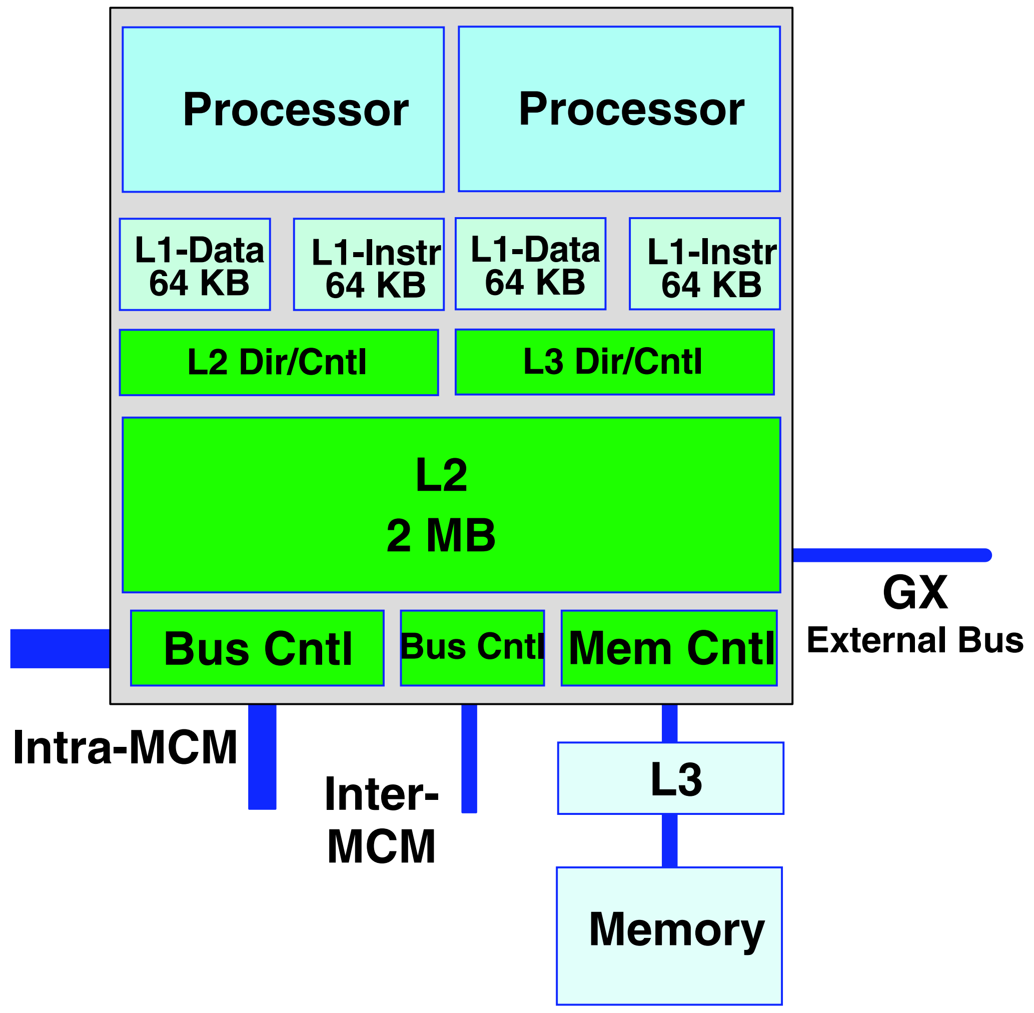
Power 3 ++

- With both Copper and Silicon
- Clock rates 20-30% faster
- Larger Caches??

Future Architectures?

- Architectures for the next 2-3 years will be clusters of SMP nodes
- Architectures will move to NUMA
- User will have a challenge to get close to CPU GFLOP numbers
- Some existing programs will run well
- Some existing programs will not run well

GP Chip

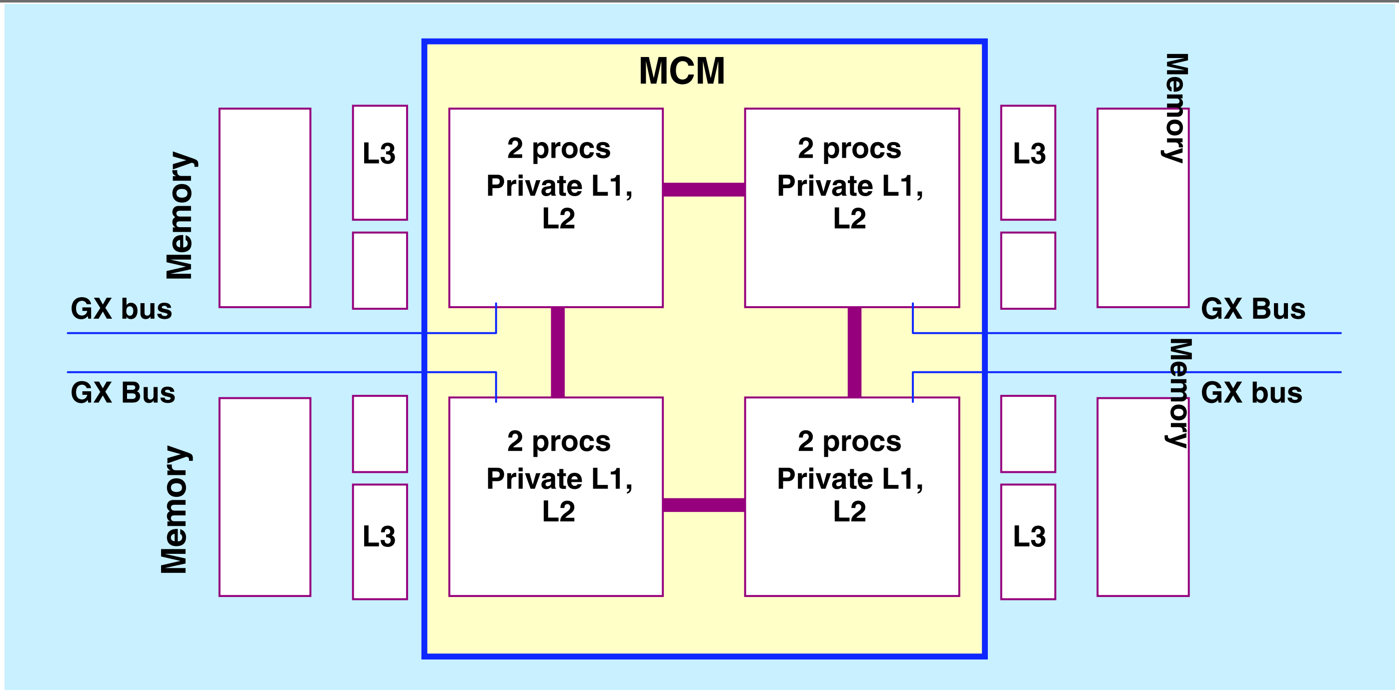


- **2 Processors per chip**
  - Advanced superscalar
  - Out-of-order execution
  - Enhanced branch prediction
  - 7 execution Units
  - Multiple outstanding miss support and prefetch logic
- **Private on-chip L1 caches**
- **Large on-chip L2 shared between 2 processors**
- **Large L3 shared between all processsors in node**
  - Up to 32 MB per GP chip
- **Large shared memory**
  - Up to 32 GB/ GP chip
- **Multiple, dedicated, high-bandwidth buses**

Gigaprocessor

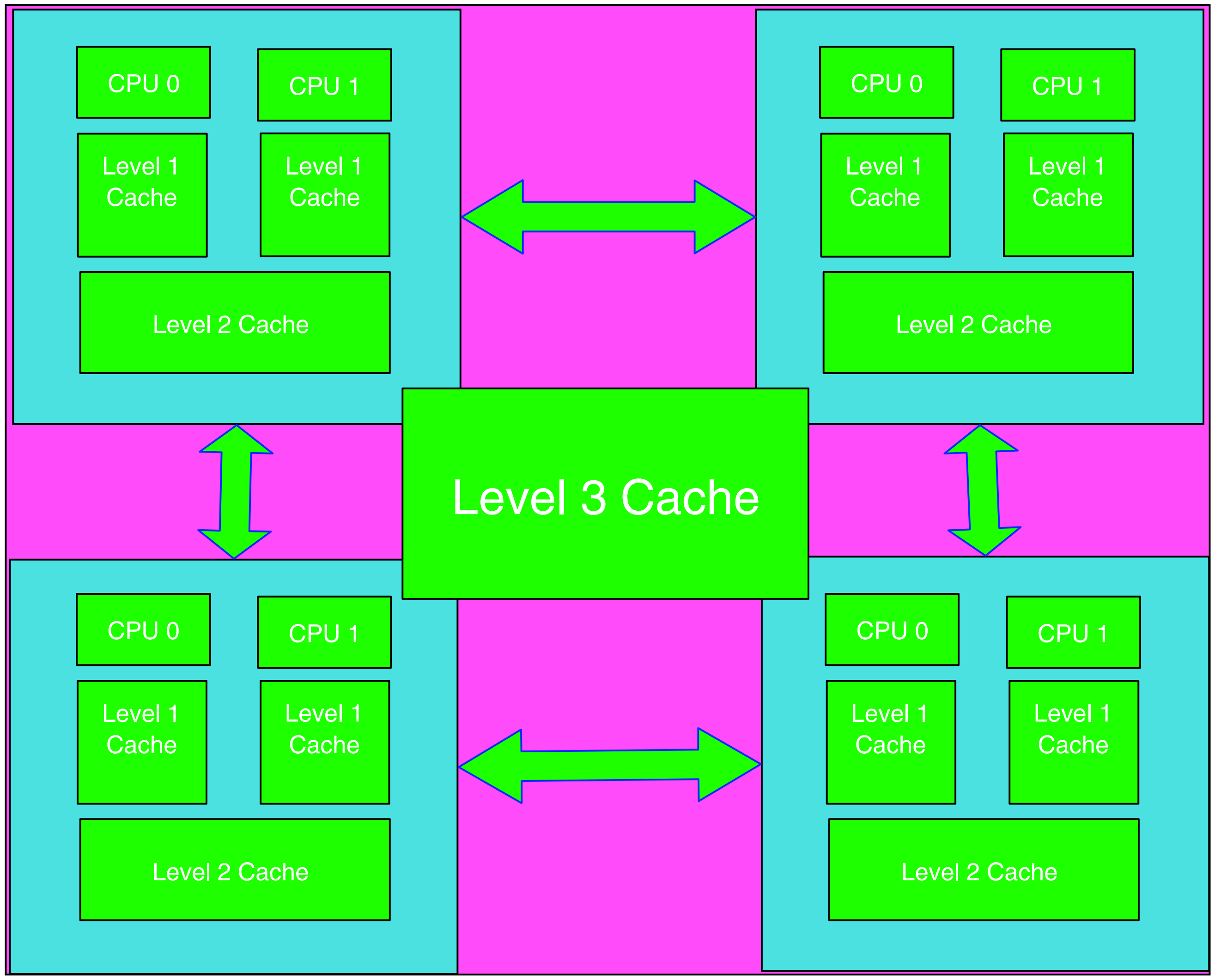
- 2 FMAs
- TLB handles larger page sizes
- > 1000 MHZ

GP 8-way

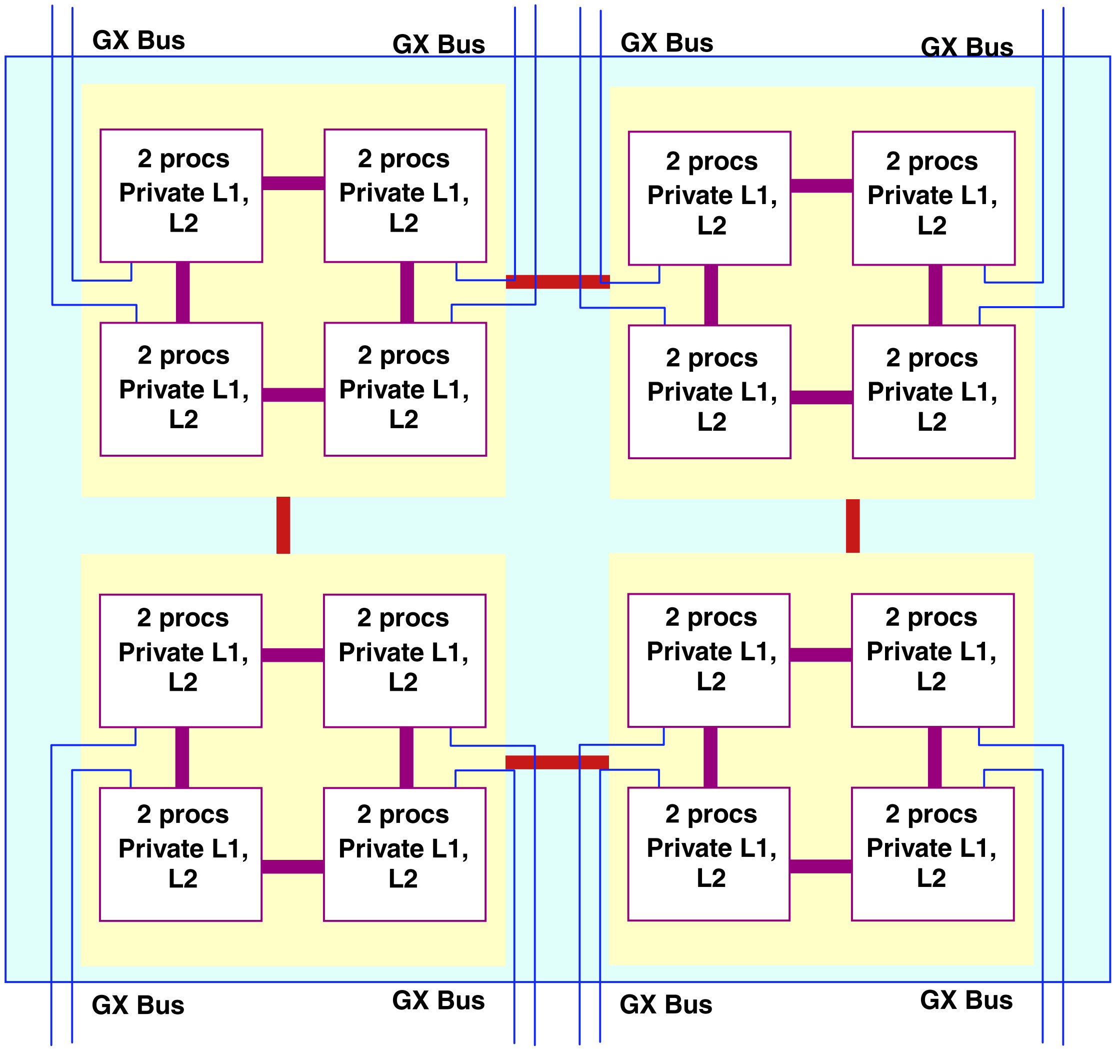


- 4 GP chips (8 processors) on an MCM
- Logically shared L3 cache
- Logically UMA
- 4 GX links for external connections
- SP Thin / Wide Node

Memory Architecture of the Gigaprocessor Module



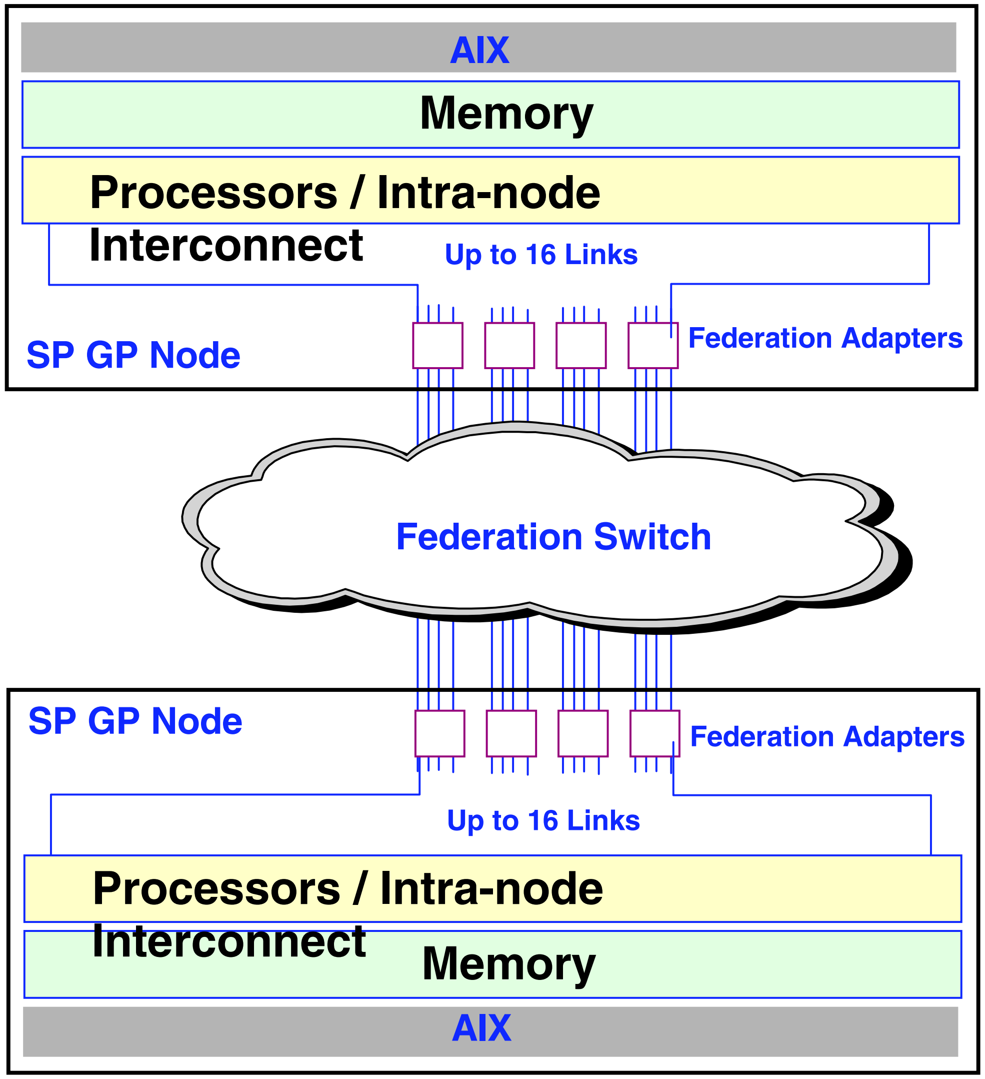
GP 32-way



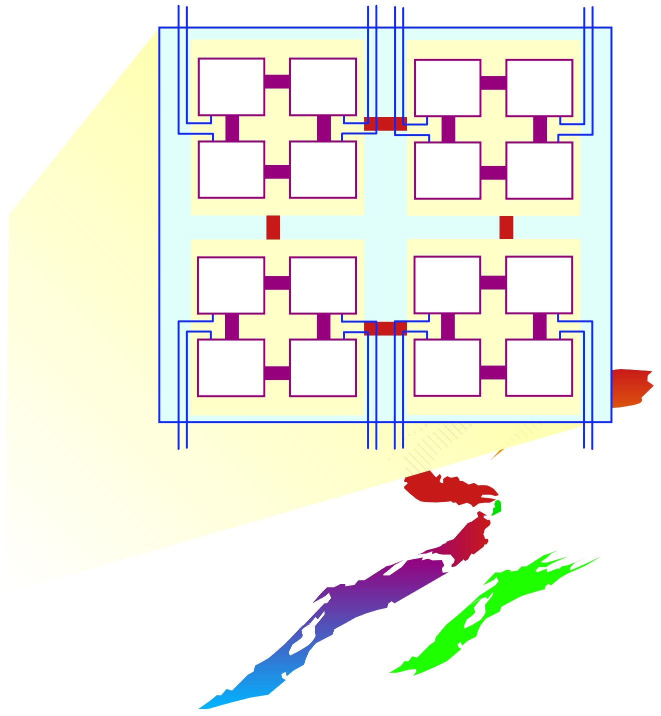
Logical UMA

SP High Node

GP-based SP System



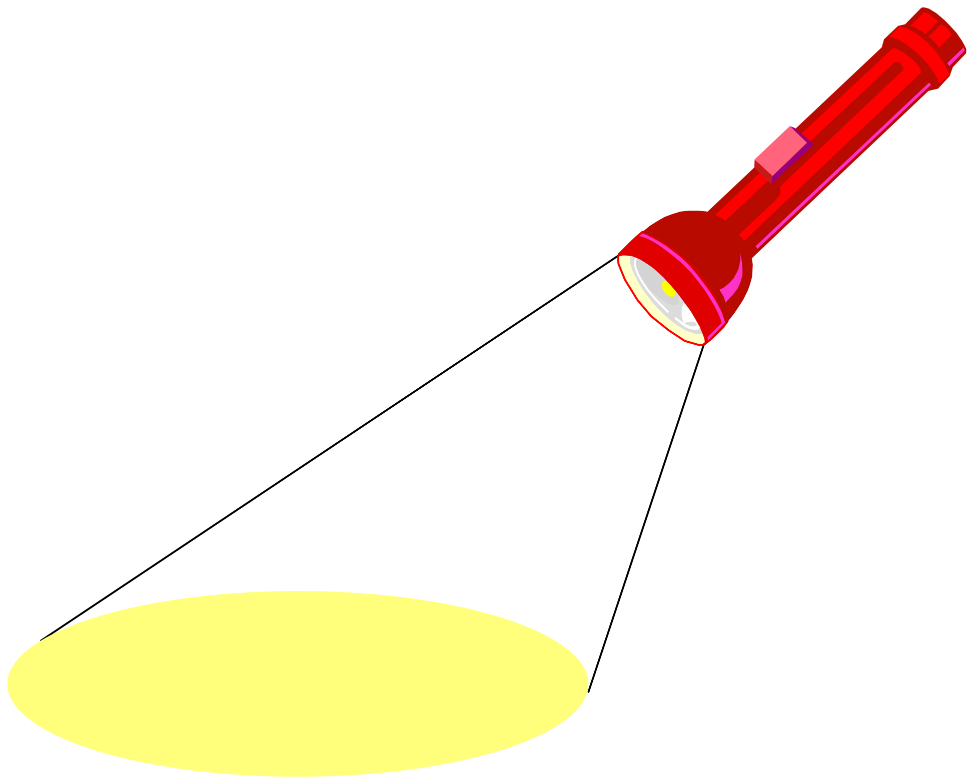
- 32 way GP High node
- Own copy of AIX
- 128+ GFLOPS/high node
- Multiple Federation Adapters for scaleable inter-node BW



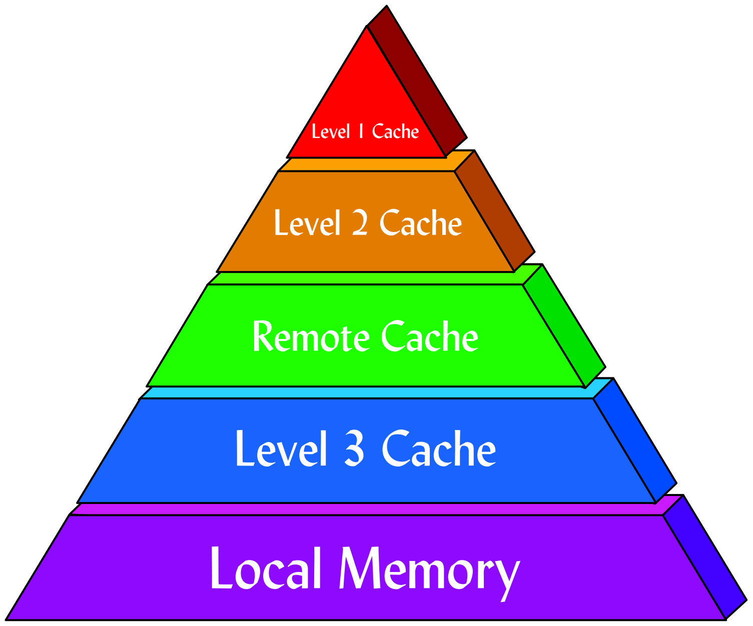
Getting to a Petaflop

- 2 Gigahertz processors generating 4 floating point operations/clock cycle
  - 8 Gigaflops
- 4 Chips on a module
  - 32 Gigaflops
- 32 processor on a board - SMP
  - 128 Gigaflops
- 8 Nodes > Teraflop
- 8192 Nodes > Petaflop

Where's the Memory??



Memory Hierarchies on the Gigaprocessor



- Level 1 Cache
  - On Chip
- Level 2 Cache
  - On Chip
- Remote (L2) Caches
- Level 3 Cache
  - Big and on Module
- Local Memory

Memory Hierarchy Research

- Example: Recursive Linear Algebra
  - Matrix multiplication at 95% of peak
  - 750 MFlops on Power3 (800 MF max)
  - Similar for other routines (solvers and factorizations).
- New Data Format
  - Recursive data storage.
  - Vast improvement over "column" and "row" storage.
  - Self-blocking at all levels of memory hierarchy.
- "Pre"-ESSL in progress.

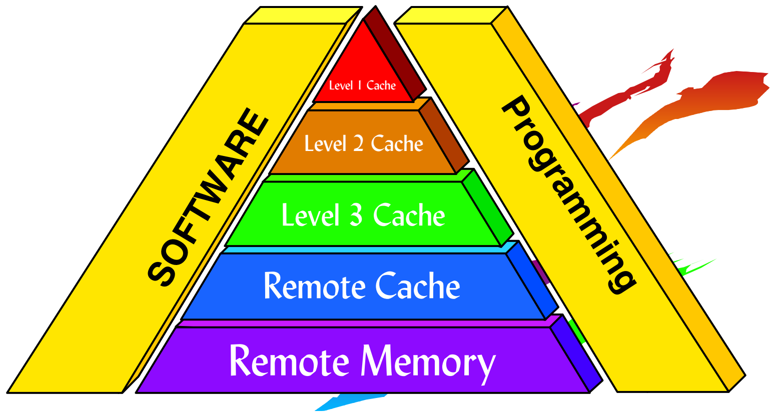
Software Challenges

Scalable Shared Memory?

Scalable Shared Memory

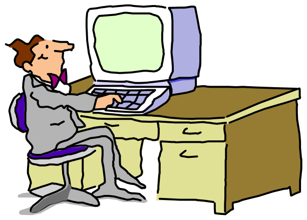
- User has a single address space
  - Compiler should be smart about cache locality
  - Other processor's cache is not far away

- Difficult Problem
  - User is smart when programming SSM



Programming Challenges

Data Locality



User Must Have

- Intelligent Compilers
  - User Input for asserting important runtime information
- Useful Tools
  - Simulation of memory hierarchy
  - Performance measurement of execution
  - Memory Mapping Tools

User Must Understand

- How program accesses memory
- How to advise the compiler
  - Directives are nice
- Organize data to be accessible in cache lines

Ideally

- Users won't expect AUTOMAGIC Optimization
- Compilers will be able to force cache residency
- Users will be able to write highly efficient programs